Package 'MARSS'

October 19, 2010

Type Package

Title Multivariate Autoregressive State-Space Modeling

Version 1.1

Date 2010-10-01

Depends MASS, mvtnorm, nlme, time

Author Eli Holmes, Eric Ward, and Kellie Wills, NOAA, Seattle, USA

Maintainer eli.holmes@noaa.gov <eli.holmes@noaa.gov>

Description The MARSS package fits constrained and unconstrained linear multivariate autoregressive state-space (MARSS) models to multivariate time series data. Type show.doc(MARSS, manual) at the R command line to open the MARSS User Guide.

License GPL-2

LazyLoad yes

LazyData yes

Repository CRAN

Date/Publication 2010-10-19 07:31:06

R topics documented:

MARSS-package 2
allowed 4
checkPopWrap 4
CSEGriskfigure 5
CSEGtmufigure 7
find.degenerate 8
graywhales 9
harborSeal 10
is.blockdiag 11

loggerhead . . . 12
MARSS . . . 13
MARSSaic . . . 19
MARSSapplynames . . . 20
MARSSboot . . . 22
MARSScheckdims . . . 24
MARSShessian . . . 25
MARSSinits . . . 26
MARSSinnovationsboot . . . 27
MARSSkem . . . 29
MARSSkemcheck . . . 33
MARSSkf . . . 34
MARSSLLprofile . . . 36
marssm . . . 38
marssm-class . . . 40
MARSSmcinit . . . 40
marssMLE . . . 42
marssMLE-class . . . 44
MARSSoptim . . . 44
MARSSoptions . . . 47
MARSSparamCIs . . . 48
MARSSsimulate . . . 49
MARSSvectorizeparam . . . 50
plankton . . . 51
popWrap . . . 52
popWrap-class . . . 55
show.doc . . . 55
stdInnov . . . 56

Index **58**

`MARSS-package` *Multivariate Autoregressive State-Space Model Estimation*

Description

The MARSS package fits constrained and unconstrained multivariate autoregressive time-series models to multivariate time series data. To open the manual from the command line, type `show.doc(MARSS, manual)`. To open an overview page with package information, type `show.doc(MARSS, index)`.

The MARSS model is

x(t+1) = B x(t) + U + w(t), where w(t) ~ MVN(0,Q)

y(t) = Z x(t) + A + v(t), where v(t) ~ MVN(0,R)

x(1) ~ MVN(x0, V0)

The parameters, hidden state processes (x), and observations (y) are matrices:

- x(t) is m x 1
- y(t) is n x 1 (m<=n)
- Z is n x m
- B is m x m
- U is m x 1
- Q is m x m
- A is n x 1
- R is n x n
- x0 is m x 1
- V0 is m x m

The package functions estimate the parameters U, Q, A, R, and x0 using a Kalman-EM algorithm (primarily but see `MARSSoptim`). Parameters may be constrained to have shared elements (elements which are constrained to have the same value) or fixed elements (with the other elements estimated). The states and smoothed state estimates are provided via a Kalman filter and smoother. Bootstrapping, confidence interval estimation, bias estimation, model selection and simulation functions are provided. The main user interface to the package is the top-level function `MARSS`.

Details

Important MARSS functions:

`MARSS` Top-level function for specifying and fitting MARSS models.

`MARSSsimulate` Produces simulated data from a MARSS model.

`MARSSkem` Estimates MARSS parameters using an Kalman-EM algorithm.

`MARSSkf` Kalman filter and smoother.

`MARSSoptim` Estimates MARSS parameters using a quasi-Newton algorithm via `optim`.

`MARSSaic` Calculates AICc, AICc, and various bootstrap AICs.

`MARSSboot` Creates bootstrap MARSS parameter estimates.

`MARSSparamCIs` Computes confidence intervals for maximum-likelihood estimates of MARSS parameters.

Author(s)

Eli Holmes, Eric Ward and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov,

kellie(dot)wills(at)noaa(dot)gov

References

The MARSS manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112.

Type `show.doc(MARSS, manual)` at the R command line to open the MARSS Manual.

Type `show.doc(MARSS, index)` to see all the package documentation, tutorials, and case study scripts.

allowed *MARSS function defaults and allowed methods*

Description

Defaults and allowed fitting methods for the MARSS function are specified in the file MARSSsettings.R. These are hidden thus to see them preface with MARSS:::.

Details

allowed is a list with the allowed constraints for each fitting method (used in checkPopWrap. allowed.methods is a vector with the allowed method arguments for the MARSS function. kem.methods and optim.methods are vectors of method arguments that fall in each of these two categories; used by MARSS. alldefaults is a list that specifies the defaults for MARSS arguments if they are not passed in; used by checkPopWrap. model.elem and model.elem.w.V0 specify the parameters names used in lists such are par lists.

checkPopWrap *Check Arguments to popWrap()*

Description

Checks inputs for wrapper object (popWrap) creation to ensure that the wrapper object can be handled by as.marssm. This is a utility function in the MARSS-package.

Usage

```
checkPopWrap(wrapperObj, wrapper.el, allowed, silent=FALSE)
```

Arguments

wrapperObj	An object of class popWrap.
wrapper.el	Wrapper elements; generally set by MARSS
allowed	Allowed constraints. This changes depending on the fitting method the user has specified (in MARSS or in the marssMLE object). Lists of for allowed for different fitting methods is set in MARSSsettings.R. Use MARSS:::allowed to view since allowed is hidden.
silent	Suppresses errors and warnings printing.

Details

Called by popWrap to ensure that user inputs are valid and can be handled by as.marssm.

Value

TRUE if object passes all checks.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

popWrap as.marssm

Examples

```
## Not run:
## Error:
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
wrapperObj = popWrap(dat, allowed=MARSS:::allowed$kem, constraint=list(Z="wrong"))

## End(Not run)
```

CSEGriskfigure *Plot Extinction Risk Metrics*

Description

Generates a six-panel plot of extinction risk metrics used in Population Viability Analysis (PVA). This is a function used by one of the vignettes in the MARSS-package.

Usage

```
CSEGriskfigure(data, te = 100, absolutethresh = FALSE, threshold = 0.1,
  datalogged = FALSE, silent = FALSE, return.model = FALSE,
  CI.method = "hessian", CI.sim = 1000)
```

Arguments

data	A data matrix with 2 columns; time in first column and counts in second column. Note time is down rows, which is different than the base MARSS-package functions.
te	Length of forecast period (positive integer)
absolutethresh	Is extinction threshold an absolute number? (T/F)
threshold	Extinction threshold either as an absolute number, if absolutethresh=TRUE, or as a fraction of current population count, if absolutethresh=FALSE.

datalogged	Are the data already logged? (T/F)
silent	Suppress printed output? (T/F)
return.model	Return state-space model as marssMLE object? (T/F)
CI.method	Confidence interval method: "hessian", "parametrc", "innovations", or "none". See MARSSparamCIs.
CI.sim	Number of simulations for bootstrap confidence intervals (positive integer).

Details

Panel 1: Time-series plot of the data. Panel 2: CDF of extinction risk. Panel 3: PDF of time to reach threshold. Panel 4: Probability of reaching different thresholds during forecast period. Panel 5: Sample projections. Panel 6: TMU plot (uncertainty as a function of the forecast).

Value

If return.model=TRUE, an object of class marssMLE.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accesses via show.doc(MARSS, manual)

(theory behind the figure) Holmes, E. E., J. L. Sabo, S. V. Viscido, and W. F. Fagan. (2007) A statistical approach to quasi-extinction forecasting. Ecology Letters 10:1182-1198.

(CDF and PDF calculations) Dennis, B., P. L. Munholland, and J. M. Scott. (1991) Estimation of growth and extinction parameters for endangered species. Ecological Monographs 61:115-143.

(TMU figure) Ellner, S. P. and E. E. Holmes. (2008) Resolving the debate on when extinction risk is predictable. Ecology Letters 11:E1-E5.

See Also

MARSSboot marssMLE CSEGtmufigure

Examples

```
d = harborSeal[,1:2]
kem = CSEGriskfigure(d,  datalogged = TRUE)
```

`CSEGtmufigure` *Plot Forecast Uncertainty*

Description

Plot the uncertainty in the probability of hitting a percent threshold (quasi-extinction) for a single random walk trajectory. This is the quasi-extinction probability used in Population Viability Analysis. The uncertainty is shown as a function of the forecast, where the forecast is defined in terms of the forecast length (number of time steps) and forecasted decline (percentage). This is a function used by one of the vignettes in the `MARSS-package`.

Usage

```
CSEGtmufigure(N = 20, u = -0.1, s2p = 0.01, make.legend = TRUE)
```

Arguments

`N`	Time steps between the first and last population data point (positive integer)
`u`	Per time-step decline (-0.1 means a 10% decline per time step; 1 means a doubling per time step.)
`s2p`	Process variance (Q). (a positive number)
`make.legend`	Add a legend to the plot? (T/F)

Details

This figure shows the region of high uncertainty in dark grey. In this region, the minimum 95 percent confidence intervals on the probability of quasi-extinction span 80 percent of the 0 to 1 probability. Green hashing indicates where the 95 percent upper bound does not exceed 5% probability of quasi-extinction. The red hashing indicates, where the 95 percent lower bound is above 95% probability of quasi-extinction. The light grey lies between these two certain/uncertain extremes. The extinction calculation is based on Dennis et al. (1991). The minimum theoretical confidence interval is based on Fieberg and Ellner (2000). This figure was developed in Ellner and Holmes (2008).

Examples using this figure are shown in the manual (`show.doc(MARSS, manual)`) in the PVA case study.

Author(s)

Eli Holmes, NOAA, Seattle, USA, and Steve Ellner, Cornell Univ.

eli(dot)holmes(at)noaa(dot)gov

References

Dennis, B., P. L. Munholland, and J. M. Scott. (1991) Estimation of growth and extinction parameters for endangered species. Ecological Monographs 61:115-143.

Fieberg, J. and Ellner, S.P. (2000) When is it meaningful to estimate an extinction probability? Ecology, 81, 2040-2047.

Ellner, S. P. and E. E. Holmes. (2008) Resolving the debate on when extinction risk is predictable. Ecology Letters 11:E1-E5.

See Also

CSEGriskfigure

Examples

```
CSEGtmufigure(N = 20, u = -0.1, s2p = 0.01)
```

`find.degenerate` *Find degenerate variance parameters*

Description

A helper function to find degenerate variance parameters in Kalman-EM estimates in the package MARSS-package.

Usage

```
find.degenerate(MLEobj)
```

Arguments

MLEobj	An object of class marssMLE as output by MARSSkem. Typically after a call to MARSS: `MARSS(data,method="kem")`.

Details

This function plots the log of the absolute value of the variance element (on the diagonal) against the log iteration number. Such log-log plots are commonly used to assess convergence in iterative routines. In state-space models with both process and non-process (observation) variance, it is entirely possible that the highest likelihood occurs when one of the variance element is zero. In this case, that element is degenerate (=0). Since the likelihood computations will generate an error, this means you will not be able to compute the true likelihood. Technically, it is certainly possible to compute the likelihood, but the code in MARSSkf will throw an error if you try to set one of the variances to zero.

Value

A plot where each diagonal variance element appears in a panel. Converged elements with have a flat log-log plot. Degenerate variance elements will have a declining log-log plot. Typically, degenerate variance parameters will show a slope of ca -1.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

See Also

MARSSkem

Examples

```
dat = t(harborSealnomiss)
#shorten the data set to 10 yrs in order to produce a degenerate solution
#in this case there is not enough data to estimate all the Q variances
dat = dat[2:4,1:10]
#For example sake, it's stopped at 50 interations as it could go on for a long time trying
MLEobj = MARSS(dat, control=list(maxit=50))
find.degenerate(MLEobj)
```

---

graywhales *Population Data Sets*

Description

Example data sets for use in MARSS PVA vignettes in the MARSS-package manual. All are UNLOGGED population counts. The data sets are matrices with year in the first column and counts in other columns. Since MARSS functions require time to be across columns, these data matrices must be transposed before passing into MARSS functions.

Usage

```
data(graywhales)
data(grouse)
data(isleRoyal)
data(prairiechicken)
data(wilddogs)
```

Format

The data are supplied as a matrix with years in the first column and counts in the second (and third for isleRoyal) columns.

Source

- graywhales Gerber L. R., Master D. P. D. and Kareiva P. M. (1999) Gray whales and the value of monitoring data in implementing the U.S. Endangered Species Act. Conservation Biology, 13, 1215-1219.
- grouse Hays D. W., Tirhi M. J. and Stinson D. W. (1998) Washington state status report for the sharptailed grouse. Washington Department Fish and Wildlife, Olympia, WA. 57 pp.
- isleRoyal Peterson R. O., Thomas N. J., Thurber J. M., Vucetich J. A. and Waite T. A. (1998) Population limitation and the wolves of Isle Royale. In: Biology and Conservation of Wild Canids (eds. D. Macdonald and C. Sillero-Zubiri). Oxford University Press, Oxford, pp. 281-292.

- prairiechicken Peterson M. J. and Silvy N. J. (1996) Reproductive stages limiting productivity of the endangered Attwater's prairie chicken. Conservation Biology, 10, 1264-1276.
- wilddogs Ginsberg, J. R., Mace, G. M. and Albon, S. (1995). Local extinction in a small and declining population: Wild Dogs in the Serengeti. Proc. R. Soc. Lond. B, 262, 221-228.

Examples

```
str(graywhales)
str(grouse)
str(isleRoyal)
str(prairiechicken)
str(wilddogs)
```

`harborSeal` *Harbor Seal Population Count Data (Log counts)*

Description

Data sets used in MARSS vignettes in the `MARSS-package`. These are data sets based on LOGGED count data from Oregon, Washington and California sites where harbor seals were censused while hauled out on land. "harborSeallnomiss" is an extrapolated data set where missing values in the original dataset have been extrapolated so that the data set can be used to demonstrate fitting population models with different underlying structures.

Usage

```
data(harborSeal)
data(harborSealnomiss)
data(harborSealWA)
```

Format

Matrix "harborSeal" contains columns "Years", "StraitJuanDeFuca", "SanJuanIslands", "EasternBays", "PugetSound", "HoodCanal", "CoastalEstuaries", "OlympicPeninsula", "CA.Mainland", "OR.NorthCoast", "CA.ChannelIslands", and "OR.SouthCoast". Matrix "harborSealnomiss" contains columns "Years", "StraitJuanDeFuca", "SanJuanIslands", "EasternBays", "PugetSound", "HoodCanal", "CoastalEstuaries", "OlympicPeninsula", "OR.NorthCoast", and "OR.SouthCoast". Matrix "harborSealWA" contains columns "Years", "SJF", "SJI", "EBays", "PSnd", and "HC", representing the same five sites as the first five columns of "harborSeal".

Details

Matrix "harborSealWA" contains the original 1978-1999 LOGGED count data for five inland WA sites. Matrix "harborSealnomiss" contains 1975-2003 data for the same sites as well as four coastal sites, where missing values have been replaced with extrapolated values. Matrix "harborSeal" contains the original 1975-2003 LOGGED data (with missing values) for the WA and OR sites as well as a CA Mainland and CA ChannelIslands time series.

Source

Jeffries et al. 2003. Trends and status of harbor seals in Washington State: 1978-1999. Journal of Wildlife Management 67(1):208-219.

Examples

```
str(harborSealWA)
str(harborSealnomiss)
str(harborSeal)
```

`is.blockdiag` *Matrix Utilities*

Description

Matrix utilities for MARSS functions in the `MARSS-package`.

Usage

```
is.blockdiag(x)
is.blockequaltri(x, uniqueblocks=FALSE)
is.blockunconst(x, uniqueblocks=FALSE)
is.diagonal(x)
is.equaltri(x)
makediag(x, nrow=NA)
takediag(x)
is.design(x)
is.fixed(x)
is.identity(x)
vec(x)
unvec(x, dim=NULL)
is.wholenumber(x, tol = .Machine$double.eps^0.5)
as.design(fixed,free)
Imat(x)
```

Arguments

`x`	A matrix (or vector for '`makediag`').
`dim`	Matrix dimensions.
`fixed`	A fixed matrix per the MARSS specification for fixed matrix syntax.
`free`	A free matrix per the MARSS specification for free matrix syntax.
`nrow`	Number of rows.
`tol`	Tolerance.
`uniqueblocks`	Must blocks be unique?

Details

'is...' tests for various matrix properties. vec(x) creates a column vector from a matrix per the standard vec math function. unvec(c,dim) takes the vector c and creates a matrix with the specified dimensions. as.design(fixed,free) returns the fixed vector and design matrix for a fixed/free pair. Imat(nrow) returns the identity matrix of dimension nrow.

Value

'makediag(x)': a matrix with diagonal x. 'takediag(x)': the diagonal from matrix x.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

---

loggerhead *Loggerhead Turtle Tracking Data*

Description

Data used in MARSS vignettes in the MARSS-package. Tracking data from ARGOS tags on eight individual loggerhead turtles, 1997-2006.

Usage

```
data(loggerhead)
data(loggerheadNoisy)
```

Format

Data frames "loggerhead" and "loggerheadNoisy" contain the following columns:

turtle Turtle name.

day Day of the month (character).

month Month number (character).

year Year (character).

lon Longitude of observation.

lat Latitude of observation.

Details

Data frame "loggerhead" contains the original latitude and longitude data. Data frame "loggerhead-Noisy" has noise added to the lat and lon data to represent data corrupted by errors.

Source

Gray's Reef National Marine Sanctuary (Georgia) and WhaleNet: http://whale.wheelock.edu/whalenet-stuff/stop_cover_archive.html

Examples

```
str(loggerhead)
str(loggerheadNoisy)
```

MARSS *Interface MARSS Model Specification and Estimation*

Description

A top-level MARSS-package function to perform model specification and estimation for multivariate autoregressive state-space (MARSS) models. To open the manual from the command line, type `show.doc(MARSS, manual)`. To open an overview page with package information, type `show.doc(MARSS, index)`.

MARSS models take the form:

x(t+1) = B x(t) + U + w(t), where w(t) ~ MVN(0,Q)

y(t) = Z x(t) + A + v(t), where v(t) ~ MVN(0,R)

x(1) ~ MVN(x0, V0)

MARSS provides an interface to the base MARSS-package functions so that users do not need to directly construct marssm and marssMLE objects.

Usage

```
MARSS(y,
    inits=NULL,
    constraint=NULL,
    fixed=NULL, free=NULL,
    miss.value=-99,
    method = "kem",
    fit=TRUE,
    silent = FALSE,
    control = NULL)
```

Arguments

	The default settings for the optional arguments are set in `MARSSsettings.R` and are given below in the details section.
	A n x T matrix of n time series over T time steps.
`inits`	List with up to 7 matrices A, R, B, U, Q, x0, V0, specifying initial values for parameters.

- B Initial value(s) for B matrix (length 1 or m x m)
- U Initial value(s) for U matrix (length 1 or m x 1)
- Q Initial value(s) for process error variance(s) (length 1 or m x m)
- A Initial value(s) for observation bias (length 1 or n x 1)
- R Initial value(s) for non-process (observation) error variance(s) (length 1 or n x n)
- x0 Initial value(s) for hidden state(s) at time=1 (length 1 or m x 1)
- V0 Initial variance(s) for hidden state(s) at time=1 (length 1 or m x m)

`constraint` Model specification using parameter constraint descriptions. See Details.

`fixed` Optional model specification using matrices of fixed and free parameters. See Details.

`free` Optional model specification using matrices of fixed and free parameters. See Details.

`miss.value` How are missing values represented in the data?

`method` Estimation method. MARSS 1.0 allows method="kem" and "BFGS".

`fit` TRUE/FALSE Whether to fit the model to the data. If FALSE, a marssMLE object with only the model is returned.

`silent` TRUE/FALSE Suppresses printing of full error messages, warnings, progress bars and convergence information.

`control` Estimation options for the maximization algorithm. The control options for method="kem" are

- `minit` The minimum number of iterations to do in the maximization routine (if needed by method). If `method="kem"`, this is an easy way to up the iterations and see how your estimates are converging. (positive integer)
- `maxit` Maximum number of iterations to be used in the maximization routine (if needed by method) (positive integer).
- `abstol` Convergence tolerance for the maximization routine. (default is 0.01 which is a bit high.)
- `iter.V0` The value of V0 to be used in place of 0 when x0 is treated as fixed and V0=0. See manual for discussion of initial state variance. (default is 10 which works well)
- `trace` A positive integer. If not 0, a record will be created during maximization iterations (what's recorded depends on the method of maximization).
- `MCInit` If TRUE, do a Monte Carlo search of the initial condition space. (T/F)
- `numInits` Number of random initial value draws if MCInit=TRUE (ignored otherwise). (positive integer)
- `numInitSteps` Number of EM iterations for each random initial value draw if MCInit=TRUE (ignored otherwise). (positive integer)
- `boundsInits` Bounds on the uniform distributions from which initial values will be drawn if MCInit=TRUE (ignored otherwise).

- `silent` 1 or TRUE, Suppresses all printing including progress bars, error messages and convergence information. 0, Turns on all printing of progress bars, fitting information and error messages. 2, Prints a brief success/failure message.

Details

MARSS provides an interface to the base `MARSS-package` functions and allows specification and fitting of MARSS models. In `MARSS-package` 1.0, the available estimation methods are maximum-likelihood via a Kalman-EM algorithm (`method="kem"`) or via a quasi-Newton algorithm provided by function `optim` (`method="BFGS"`). The function `MARSS()` allows the user to specify models using text strings for common classes of parameter matrices via the argument `constraint`. It allows the user to specify fixed values for matrices by passing in numeric matrices in the `constraint` list. If the model classes available via the `constraint` strings are not sufficient, MARSS also allows specification using matrix pairs specified with argument `fixed` and `free`. If `fixed/free` matrices are specified for some parameters, these will override any constraints for those parameters. See `marssm` or the manual (`show.doc(MARSS, manual)`) for documentation and instructions on specifying fixed and free matrices.

Valid constraints for `method="kem"` are below. See the manual (`show.doc(MARSS, manual)`) for details and type `allowed$kem` to see the allowed list specified in `MARSSsettings.R`.

- `Z` "identity" or a vector of factors specifying which of the m hidden state time series correspond to which of the n observation time series. May also be specified as a numeric n x m matrix to use a custom fixed Z.
- `B` "identity" or a vector of factors specifying shared diagonal elements. May also be specified as a numeric m x m matrix to use custom fixed B, but in this case all the eigenvalues of B must fall in the unit circle.
- `U` "unconstrained", "equal", "unequal" or "zero". May also be a vector of factors specifying shared u terms. May also be specified as a numeric m x 1 matrix to use a custom fixed U. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated.
- `Q` "unconstrained", "diagonal and unequal", "diagonal and equal", or "equalvarcov". May also be a vector of factors specifying shared diagonal elements. May also be specified as a numeric m x m matrix to use a custom fixed Q. If the matrix is diagonal (off-diagonals all zeros), then NAs may appear on the diagonal to allow some diagonal elements to be fixed while other elements (the NAs) are estimated.
- `A` "scaling" This treats A as an intercept or "zero" which sets A to a fixed value of all zeros. May also be specified as a numeric n x 1 matrix to use a custom fixed A. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated. Note all NAs in A would mean all A elements are estimated and would typically result in an under-constrained and unsolveable model; at least one A element per X state needs to be fixed.
- `R` "unconstrained", "diagonal and unequal", "diagonal and equal", or "equalvarcov". May also be a vector of factors specifying shared diagonal elements. May also be specified as a numeric n x n matrix to use a custom fixed R. If the matrix is diagonal (off-diagonals all zeros), then NAs may appear on the diagonal to allow some diagonal elements to be fixed while other elements (the NAs) are estimated.

- x0 "unconstrained", "equal", "unequal" or "zero". May also be a vector of factors specifying shared initial state values. May also be specified as a numeric m x 1 matrix to use a custom fixed x0. NAs can be put in this matrix to allow some elements to be fixed and others (the NAs) to be estimated.

Valid constraints for method="BFGS" are the same as for method="kem" except that the Q and R matrices must be diagonal if estimated. Thus "unconstrained" and "equalvarcov" are not allowed for Q or R. Type allowed$BFGS to see the allowed list specified in MARSSsettings.

The default estimation method, method="kem", is the "Kalman-EM" algorithm described in the manual. The default settings for the optional arguments are set via alldefaults$kem in MARSSsettings. For this method, they are:

- inits = list(B=1, U=0, Q=0.05, A=0, R=0.05, x0=-99, V0=10)
- constraint = list(Z="identity", A="scaling", R="diagonal and equal", B="identity", U="unconstrained", Q="diagonal and unequal", x0="unconstrained")
- miss.value = -99
- control=list(minit=1, maxit=5000, abstol=0.01, iter.V0=10, trace=0, safe=FALSE, MCInit=FALSE, numInits = 500, numInitSteps = 10, boundsInits=list(B=c(0,1), U=c(-1,1), logQ = c(log(1.0e-05),log(1.0)), Z=c(0,1), A=c(-1,1), logR = c(log(1.0e-05),log(1.0))))

For method="BFGS", type alldefaults$BFGS to see the defaults.

The likelihood surface for MARSS models can be highly multimodal. It is recommended that for final analyses the ML estimates are checked by using the Monte Carlo initial conditions search using MCInit=TRUE in the control list. This requires more computation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs.

Value

An object of class marssMLE with the following components:

model	MARSS model specification (an object of class marssm).
start	List with 8 matrices A, R, B, U, Q, Z, x0, V0, specifying initial values for parameters.
control	A list of estimation options, as specified by arguments control.
method	Estimation method.

If fit=TRUE, the following are also added to the marssMLE object. If fit=FALSE, an marssMLE object ready for fitting via the specified method is returned.

par	A list with 8 matrices of estimated (+ fixed) parameter values Z, A, R, B, U, Q, x0, V0.
kf	A list containing Kalman filter/smoother output from MARSSkf.
numIter	Number of iterations required for convergence.
convergence	Convergence status. 0 means converged successfully. Anything else is a warning or error. 2 means the MLEobj has an error; the MLEobj is returned so you can debug it. The other numbers are errors during fitting. The error code depends on the fitting method. See MARSSkem and MARSSoptim.

logLik	Log-likelihood.
AIC	Akaike's Information Criterion.
AICc	Sample size corrected AIC.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, kellie(dot)wills(at)noaa(dot)gov

References

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

See Also

marssm marssMLE MARSSkem MARSSoptim MARSS-package

Examples

```
#harborSealWA is a n=5 matrix of logged population counts
dat = t(harborSealWA)
dat = dat[2:nrow(dat),] #remove the year row
#fit a model with 1 hidden state and 5 observation time series
kemfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
   R="diagonal and equal"))
kemfit$model #This gives a description of the model
print(kemfit$model) # same as kemfit$model
summary(kemfit$model) #This shows the model structure
show(kemfit$model)    #shows the raw object

#Look at the log-log convergence plots for the variances
find.degenerate(kemfit)

#Increase the number of iterations to ensure convergence
kemfit100 = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
   R="diagonal and equal"), control=list(minit=100))
find.degenerate(kemfit100)  #now check log-logconvergence of variances
#plot is flat; that's good

#fit the model using quasi-Newton algorithm
## Not run:  #takes a long time
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1)),
   R="diagonal and unequal"), method="BFGS")

## End(Not run)

#add CIs to a marssMLE object
#default uses an estimated Hessian matrix
kem.with.hess.CIs = MARSSparamCIs(kemfit100)
```

```
kem.with.hess.CIs #print with se's and CIs

#estimate CIs using a parametric bootstrap
tmp=kemfit
tmp$control$abstol=0.01
kem.with.boot.CIs = MARSSparamCIs(tmp,
   method="parametric", nboot=10)
#nboot should be more like 1000, but set low for example sake
kem.with.boot.CIs #print with se's, CIs, and bias est

#fit a model with 5 hidden states (default)
kemfit = MARSS(dat, silent=TRUE)  #suppress printing
kemfit #print information on the marssMLE object
show(kemfit) #look at the raw marssMLE object

#fit a model with 5 correlated hidden states
# with one variance and one  covariance using the delta logLik convergence test
# by setting abstol.  This is fast, but does not actually test convergence (i.e. that
# log param value versus log iteration is flat
kemfit = MARSS(dat, constraint=list(Q="equalvarcov"), control=list(abstol=0.1))

#fit a model with 5 correlated hidden states
#abstol set rather huge and many of the Q's are not converged based on log-log test
kemfit = MARSS(dat, constraint=list(Q="unconstrained"), control=list(abstol=0.1))

#fit a model with 5 independent hidden states
#where each observation time series is independent
#the hidden trajectories 1-3 & 4-5 share their U parameter
kemfit = MARSS(dat, constraint=list(U=factor(c("N","N","N","S","S"))))

#same model, but with fixed independent observation errors
kemfit = MARSS(dat, constraint=list(U=factor(c("N","N","N","S","S")),
   R=diag(0.01,5)),control=list(minit=100))

#fit a model with 2 hidden states (north and south)
#where observation time series 1-3 are north and 4-5 are south
#Make the hidden state process independent with same process var
#Make the observation errors different but independent
#Make the growth parameters (U) the same
kemfit = MARSS(dat, constraint=list(Z=factor(c("N","N","N","S","S")),
   Q="diagonal and equal",R="diagonal and unequal",U="equal"),
   control=list(minit=100))

#print the model followed by the marssMLE object
kemfit$model
kemfit

#simulate some new data from our fitted model
sim.data=MARSSsimulate(kemfit$par, nsim=10, tSteps=100)

## Not run:
#Compute bootstrap AIC for the model; this takes a long, long time
kemfit.with.AICb = MARSSaic(kemfit, output = "AICbp")
```

```
kemfit.with.AICb

## End(Not run)
```

`MARSSaic` *AIC for MARSS models*

Description

Calculates AIC, AICc, a parametric bootstrap AIC (AICbp) and a non-parametric bootstrap AIC (AICbb). This is a base function in the `MARSS-package`.

Usage

```
MARSSaic(MLEobj, output = c("AIC", "AICc"),
  Options = list(nboot = 1000, return.logL.star = FALSE,
  silent = FALSE))
```

Arguments

`MLEobj`	An object of class `marssMLE`. This object must have a `$par` element containing MLE parameter estimates from e.g. `MARSSkem()`.
`output`	A vector containing one or more of the following: "AIC", "AICc", "AICbp", "AICbb", "AICi", "boot.params". See Details.
`Options`	A list containing: • `nboot` Number of bootstraps (positive integer) • `return.logL.star` Return the log-likelihoods for each bootstrap? (T/F) • `silent` Suppress printing of the progress bar during AIC bootstraps? (T/F)

Details

When sample size is small, Akaike's Information Criterion (AIC) under-penalizes more complex models. The most commonly used small sample size corrector is AICc, which uses a penalty term of Kn/(n-K-1), where K is the number of estimated parameters. However, for time series models, AICc still under-penalizes complex models; this is especially true for MARSS models.

Two small-sample estimators specific for MARSS models have been developed. Cavanaugh and Shumway (1997) developed a variant of bootstrapped AIC using Stoffer and Wall's (1991) bootstrap algorithm ("AICbb"). Holmes and Ward (2010) developed a variant on AICb ("AICbp") using a parametric bootstrap. The parametric bootstrap permits AICb calculation when there are missing values in the data, which Cavanaugh and Shumway's algorithm does not allow. More recently, Bengtsson and Cavanaugh (2006) developed another small-sample AIC estimator, AICi, based on fitting candidate models to multivariate white noise.

When `output` includes both `"AICbp"` and `"boot.params"`, the bootstrapped parameters from `"AICbp"` will be added to `MLEobj`.

Value

Returns the marssMLE object that was passed in with additional AIC components added on top as specified in the 'output' argument.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accesses via `show.doc(MARSS, manual)`

Bengtsson, T., and J. E. Cavanaugh. 2006. An improved Akaike information criterion for state-space model selection. Computational Statistics & Data Analysis 50:2635-2654.

Cavanaugh, J. E., and R. H. Shumway. 1997. A bootstrap variant of AIC for state-space model selection. Statistica Sinica 7:473-496.

See Also

MARSSboot

Examples

```
dat = t(harborSealWA)
dat = dat[2:3,]
kem = MARSS(dat, constraint=list(Z=factor(c(1,1)),
  R="diagonal and equal"))
kemAIC = MARSSaic(kem, output=c("AIC","AICc"))
```

MARSSapplynames *Names for marssMLE Object Components*

Description

Puts names on the matrix components of marssMLE and marssm objects. This is a utility function in the MARSS-package and is not directly accessible. Use `MARSS:::MARSSapplynames()` to access.

Usage

```
MARSSapplynames(obj, Y.names = NA, X.names = NA, x0.names = NA,
  V0.names = NA, U.names = NA, A.names = NA, R.names = NA,
  Q.names = NA, B.names = NA, Z.names = NA,
  rows = TRUE, cols = TRUE)
```

Arguments

obj	An object of class marssMLE or marssm.
Y.names	Vector of names for observed time series.
X.names	Vector of names for the hidden state trajectories.
x0.names	Vector of names for MLEobjparx0, MLEobj$start$x0, MLEobj$model$fixed$x0 and MLEobj$model$free$x0.
V0.names	Names for MLEobjparV0
U.names	Names for MLEobjparU
A.names	Names for MLEobjparA
R.names	Names for MLEobjparR
Q.names	Names for MLEobjparQ
B.names	Names for MLEobjparB
Z.names	Row and col names for MLEobjparZ
rows	Add row names?
cols	Add column names to B, Q, R and V0 matrices?

Details

Default behavior will use names rownames of data (if available, and if not Y1, Y2, ...) for the Ys and all matrices that are n x ... and use X1, X2, ... for the Xs and all matrices that are m x ... .

Value

The object passed in, with row and column names on matrices as specified.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, kellie(dot)wills(at)noaa(dot)gov

See Also

marssMLE marssm

MARSSboot *Bootstrap MARSS Parameter Estimates*

Description

Creates bootstrap parameter estimates and simulated (or bootstrapped) data (if appropriate). This is a base function in the MARSS-package.

Usage

```
MARSSboot(MLEobj, nboot = 1000,
  output = "parameters", sim = "parametric",
  param.gen = "KalmanEM", control = NULL, silent = FALSE)
```

Arguments

MLEobj	An object of class marssMLE. Must have a $par element containing MLE parameter estimates.
nboot	Number of bootstraps to perform.
output	Output to be returned: "data", "parameters" or "all".
sim	Type of bootstrap: "parametric" or "innovations". See Details.
param.gen	Parameter generation method: "hessian" or "KalmanEM".
control	The options in MLEobj$control are used by default. If supplied here, must contain all of the following: max.iter Maximum number of EM iterations. tol Optional tolerance for log-likelihood change. If log-likelihood decreases less than this amount relative to the previous iteration, the EM algorithm exits. iter.V0 The value of V0 to be used in place of 0 when x0 is treated as fixed and V0=0. See manual for a discussion of the setting of diffuse priors for iterative parts of the maximization algorithms.
silent	Suppresses printing of progress bar.

Details

Approximate confidence intervals (CIs) on the model parameters can be calculated by numerically estimating the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates). The Hessian CIs are based on the asymptotic normality of ML estimates under a large-sample approximation. CIs that are not based on asymptotic theory can be calculated using parametric and non-parametric bootstrapping.

Stoffer and Wall (1991) present an algorithm for generating CIs via a non-parametric bootstrap for state-space models (sim = "innovations"). The basic idea is that the Kalman filter can be used to generate estimates of the residuals of the model fit. These residuals are then standardized and resampled and used to generate bootstrapped data using the MARSS model and its maximum-likelihood parameter estimates. One of the limitations of the Stoffer and Wall algorithm is that it

cannot be used when there are missing data, unless all data at time t are missing. An alternative approach is a parametric bootstrap (`sim = "parametric"`), in which the ML parameter estimates are used to produce bootstrapped data directly from the state-space model.

Value

A list with the following components:

`boot.params`	Matrix (number of params x nboot) of parameter estimates from the bootstrap.
`boot.data`	Array (n x t x nboot) of simulated (or bootstrapped) data (if requested and appropriate).
`model`	The `marssm` object that was passed in via `MLEobj$model`.
`nboot`	Number of bootstraps performed.
`output`	Type of output returned.
`sim`	Type of bootstrap.
`param.gen`	Parameter generation method: "hessian" or "KalmanEM".

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accesses via `show.doc(MARSS, manual)`

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. Journal of the American Statistical Association 86:1024-1033.

Cavanaugh, J. E., and R. H. Shumway. 1997. A bootstrap variant of AIC for state-space model selection. Statistica Sinica 7:473-496.

See Also

`marssMLE` `marssm` `MARSSaic`

Examples

```
#nboot is set low in these examples in order to run quickly
#normally nboot would be >1000 at least
dat = t(harborSealnomiss)
dat = dat[2:4,]
#normally one would not use the delta logLik test for convergence (abstol)
#but used here with big abstol and minit to speed up the example
kem = MARSS(dat, constraint=list(U="equal",Q="diagonal and equal"),
   control=list(minit=40,abstol=0.1))
hess.list = MARSSboot(kem, param.gen="hessian", nboot=5)
```

```
# (no missing values)
boot.list = MARSSboot(kem, output="all", sim="innovations", nboot=5)

# Bootstrap CIs for data with missing values
dat = t(harborSealWA)
dat = dat[2:4,]
kem = MARSS(dat, constraint=list(Q="diagonal and equal"),
   control=list(minit=40,abstol=0.1))
boot.list = MARSSboot(kem, output="all", sim="parametric", nboot=5)
```

`MARSScheckdims` *MARSS Utilities*

Description

This is a utility function in the `MARSS-package` for checking MARSS matrix dimensions and parameter lists.

Usage

```
MARSScheckdims(el, target, n, m)
MARSScheckpar(parList, n, m)
```

Arguments

`el`	Name of a MARSS list element ("A", "B", "Q", "R", "U", "V0", "x0", "Z").
`target`	List to be checked.
`n`	Number of time series.
`m`	Number of state processes.
`parList`	MARSS parameter list.

Value

TRUE if no problems; otherwise a message describing the problems.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

`MARSShessian` *MARSS Parameter Variance-Covariance Matrix from the Hessian Matrix*

Description

Calculates approximate parameter variance-covariance matrix and appends it to a `marssMLE` object. This is a utility function in the `MARSS-package`.

Usage

```
MARSShessian(MLEobj)
```

Arguments

`MLEobj`	An object of class `marssMLE`. This object must have a `$par` element containing MLE parameter estimates from e.g. `MARSSkem`.

Details

Uses `fdHess` from package `nlme` to numerically estimate the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates). Hessian CIs are based on the asymptotic normality of ML estimates under a large-sample approximation.

Value

`MARSShessian()` returns the `marssMLE` object passed in along with additional components `Hessian, gradient, parMean` and `parSigma` computed by the `MARSShessian` function.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

See Also

`MARSSparamCIs` `marssMLE`

Examples

```
dat = t(harborSeal)
dat = dat[2:4,]
MLEobj = MARSS(dat)
MLEobj.w.hessian = MARSShessian(MLEobj)
```

MARSSinits *Initial Values for MLE*

Description

Sets up generic starting values for parameters for maximum-likelihood estimation algorithms that use an iterative maximization routine needing starting values. Examples of such algorithms are the Kalman-EM algorithm in MARSSkem and Newton methods in MARSSoptim. This is a utility function in the MARSS-package.

Usage

```
MARSSinits(modelObj, inits=list(B=1, U=0, Q=0.05,
   A=0, R=0.05, x0=-99, V0=10))
```

Arguments

modelObj	An object of class marssm. MARSSinits uses three elements of the model object. • data The data element is used to determine n, the dimension of the y in the MARSS model. • fixed The fixed matrices are used to determine which (if any) model parameters are fixed. • miss.value Used if a linear regression through the data is used to construct inits for x0. In this case, the missing values are replaced by NA.
inits	A list of up to 8 values to construct starting matrices for each parameter in a MARSSmodel.

Details

Creates an inits parameter list for use by iterative maximization algorithms.

Defaults values for inits is supplied in MARSSsettings.R. The user can alter these and supply any of the following (m is the dim of X and n is the dim of Y in the MARSS model):

- elem=A,U A numeric vector or matrix which will be constructed into inits$elem by the command array(inits$elem),dim=c(n or m,1)). If elem is fixed in the model, any inits$elem values will be overridden and replaced with the fixed value. Default is array(0,dim=c(n or m,1)).
- elem=Q,R,B A numeric vector or matrix. If length equals the length modelObj$fixed$elem then inits$elem will be constructed by array(inits$elem),dim=dim(modelObj$fixed$elem)). If length is 1 or equals m or n then inits$elem will be constructed into a diagonal matrix by the command diag(inits$elem,m or n). If elem is fixed in the model, any inits$elem values will be overridden and replaced with the fixed value. Default is diag(0.05,m or n) for Q and R. Default is diag(1,m) for B.

- x0 If inits$x0=-99, then starting values for x0 are estimated by a linear regression through the count data assuming A=0. This will be a poor start if inits$A is not 0. If inits$x0 is a numeric vector or matrix, inits$x0 will be constructed by the command array(inits$x0),dim=c(m,1)). If x0 is fixed in the model, any inits$x0 values will be overridden and replaced with the fixed value. Default is inits$x0=-99.
- Z If Z is fixed in the model, inits$Z set to the fixed value. If Z is not fixed, then the user must supply inits$Z. There is no default.
- elem=V0 MARSSkem and MARSSoptim will override inits$V0 and use a diffuse prior (set with marssMLE$control$iter.V0 if any x0 are estimated and will use modelObj$fixed$V0 if x0 is fixed. The first case corresponds to x0 fixed but unknown and treated as an estimated parameter with V0=0. The diffuse prior for V0 is used only during the iterations of maximization to all x0 estimation and then V0 is reset to 0 for the final likelihood calculation. The second case corresponds to using a prior for the initial state. Although inits$V0 is ignored by the fitting algorithm in MARSS 1.0, it is set here for forward compatibility. If inits$V0 is a vector or matrix with length equal to the length modelObj$fixed$V0 then inits$V0 will be constructed by the command array(inits$V0,dim=c(m,m)). If length is 1 or equal to m then inits$V0 will be constructed into a diagonal matrix by the command diag(inits$V0,m). If x0 is fixed in the model, inits$V0 values will be overridden and replaced with modelObj$fixed$V0. Default is diag(10,m).

Value

A list with 8 matrices A, R, B, U, Q, x0, V0, Z, specifying initial values for parameters in a MARSS model.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

See Also

marssm MARSSkem MARSSoptim

MARSSinnovationsboot

Bootstrapped Data using Stoffer and Wall's Algorithm

Description

Creates bootstrap data via sampling from the standardized innovations matrix. This is a base function in the MARSS-package.

Usage

```
MARSSinnovationsboot(MLEobj, nboot = 1000, minIndx = 3)
```

Arguments

MLEobj	An object of class marssMLE. This object must have a $par element containing MLE parameter estimates from e.g. MARSSkem or MARSS. This algorithm may not be used if there are missing datapoints in the data.
nboot	Number of bootstraps to perform.
minIndx	Number of innovations to skip. Stoffer & Wall suggest not sampling from innovations 1-3.

Details

Stoffer and Wall (1991) present an algorithm for generating CIs via a non-parametric bootstrap for state-space models. The basic idea is that the Kalman filter can be used to generate estimates of the residuals of the model fit. These residuals are then standardized and resampled and used to generate bootstrapped data using the MARSS model and its maximum-likelihood parameter estimates. One of the limitations of the Stoffer and Wall algorithm is that it cannot be used when there are missing data, unless all data at time t are missing.

Value

A list containing the following components:

boot.states	Array (dim is m x tSteps x nboot) of simulated state processes.
boot.data	Array (dim is n x tSteps x nboot) of simulated data.
model	MARSS model ($model element of the marssMLE object.
nboot	Number of bootstraps performed.

m is the number state processes (x in the MARSS model) and n is the number of observation time series (y in the MARSS model).

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. Journal of the American Statistical Association 86:1024-1033.

See Also

stdInnov MARSSparamCIs MARSSboot

Examples

```
dat = t(harborSealnomiss)
dat = dat[2:nrow(dat),]
MLEobj = MARSS(dat, constraint=list(U="equal",Q="diagonal and equal"))
boot.obj = MARSSinnovationsboot(MLEobj)
```

`MARSSkem` *Maximum Likelihood Estimation for Multivariate Autoregressive State-Space Models*

Description

`MARSSkem()` performs maximum-likelihood estimation, using an EM algorithm for constrained and unconstrained MARSS models. This is one of the base functions in the `MARSS-package`.

Usage

```
MARSSkem(MLEobj)
```

Arguments

`MLEobj`	An object of class `marssMLE`.

Details

Objects of class `marssMLE` may be built from scatch but are easier to construct using `MARSS` with `MARSS(..., fit=FALSE)`.

Options for `MARSSkem()` may be set using `MLEobj$control`, as follows:

`MLEobj$control$minit` Minimum number of EM iterations. You can use this to force the algorithm to do a certain number of iterations. This is helpful if your soln is not converging.

`MLEobj$control$maxit` Maximum number of EM iterations.

`MLEobj$control$min.iter.conv.test` The minimum number of iterations before the log-log convergence test will be computed. If `control$maxit` is set less than this, then convergence will not be computed (and the algorithm will just run for maxit iterations).

`MLEobj$control$conv.test.deltaT` The number of iterations to use in the log-log convergence test. This defaults to 9.

`MLEobj$control$abstol` Tolerance for log-likelihood change for the delta logLik convergence test. If log-likelihood changes less than this amount relative to the previous iteration, the EM algorithm exits. This is normally (default) set to NULL and the log-log convergence test is used instead.

`MLEobj$control$iter.V0` This is the value to which V0 will be set during the EM algorithm (!=0). See Details.

`MLEobj$control$trace` A positive integer. If not 0, a record will be created of each variable over all EM iterations and detailed warning messages (if appropriate) will be printed.

`MLEobj$control$safe` If TRUE, `MARSSkem` will rerun `MARSSkf` after each individual parameter update rather than only after all parameters are updated. The latter is slower and unnecessary for many models, but in some cases, the safer and slower algorithm is needed because the ML parameter matrices have high condition numbers.

`MLEobj$control$MCInit` If TRUE, Monte Carlo initialization will be performed by `MARSSmcinit`.

`MLEobj$control$numInits` Number of random initial value draws to be used with `MARSSmcinit`. Ignored if `control$MCInit=FALSE`.

`MLEobj$control$numInitSteps` Maximum number of EM iterations for each random initial value draw to be used with `MARSSmcinit`. Ignored if `control$MCInit=FALSE`.

`MLEobj$control$boundsInits` Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn to be used with `MARSSmcinit()`. Ignored if `control$MCInit=FALSE`. See Examples.

`MLEobj$control$silent` Suppresses printing of progress bars, error messages, warnings and convergence information.

Value

The `marssMLE` object which was passed in, with additional components:

`method` String "kem".

`kf` Kalman filter output.

`iter.record` If `MLEobj$control$trace = TRUE`, a list with `par` = a record of each estimated parameter over all EM iterations and `logLik` = a record of the log likelikelihood with V0 set to the value of `iter.V0`. Note this is different than the log likelihood with V0 = 0 (which is the final log likelihood value).

`numIter` Number of iterations needed for convergence.

`convergence` Did estimation converge successfully?

convergence=0 Converged in less than `MLEobj$control$maxit` iterations and no evidence of degenerate solution using the log-log plot test.

convergence=1 Maximum number of iterations `MLEobj$control$maxit` was reached before `MLEobj$control$abstol` condition was satisfied. This value can only be output if `MLEobj$control$abstol` is passed in. If the default log-log convergence test is being used, convergence=10 is returned when maxit is reached.

convergence=2 No convergence diagnostic were computed because there are problems with the MLE object.

convergence=3 No convergence diagnostic were computed because `MLEobj$control$maxit` was set to less than `control$min.iter.conv.test`. This is not an error.

convergence=10 Some of the parameter estimates did not converge (based on the log-log plot test) before `MLEobj$control$maxit` was reached. This is not an error per se. Degenerate solutions are fine but the MARSS algorithm will not compute the proper likelihood for a degenerate solution.

convergence=11 Some of the parameter estimates did not converge (based on the log-log plot test) even though `MLEobj$control$abstol` was reached.

This value can only be output if `MLEobj$control$abstol` is passed in. Try not setting `MLEobj$control$abstol` so that the log-log convergence test is used instead. If it takes too long, try making `MLEobj$control$maxit` smaller (like 500).

convergence=12 `MLEobj$control$abstol` was reached but the log-log plot test returned NAs. This is an odd error and you should set `control$trace=TRUE` and look at the outputted `$iter.record` to see what is wrong. This value can only be output if `MLEobj$control$abstol` is passed in.

convergence=52 The algorithm was abandoned due to numerical errors. Usually this means one of the variances either went to zero or to all elements being equal. This is not an error per se. Most likely it means that your model is not very good for your data (too inflexible or too many parameters). Try setting `control$trace=1` to view a detailed error report.

convergence=62 The algorithm was abandoned due to errors in the log-log convergence test. You should not get this error (it is included for debugging purposes to catch improper arguments passed into the log-log convergence test). If you get it, try passing in `control$abstol` to use delta logLik as the convergence test.

convergence=72 Other convergence errors. This is included for debugging purposes to catch misc. errors. If you get it, try passing in `control$abstol` to use delta logLik as the convergence test.

`logLik` Log-likelihood.

`states` State estimates from the Kalman filter.

`states.se` Confidence intervals based on state standard errors, see caption of Fig 6.3 (p. 337) Shumway & Stoffer.

`errors` Any error messages.

Discussion

To ensure that the global maximum-likelihood values are found, it is recommended that initial parameter values be set using Monte Carlo initialization (`MLEobj$control$MCInit` = TRUE), particularly if the model is not a good fit to the data. This requires more compuation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs. For many models, this is unnecessary, but answers should be checked using an initial conditions search before reporting final values.

`MARSSkem()` calls a Kalman filter/smoother (`MARSSkf`) for hidden state estimation. The algorithm allows two options for the initial state conditions: x at t=1. Either x0 is treated as fixed but unknown (estimated); in this case, `fixed$V0=0` and x0 is estimated. This is the default behavior. In the second case, the initial conditions are specified with a known prior, `fixed$x0` and `fixed$V0`, and x0 is not estimated. In the first case, x0 fixed but unknown and to be estimated, V0=0 but the algorithm cannot be run with this since x0 would never move from its initial value at iteration=1. Instead, during the EM iterations, a diffuse prior is used. This is done by setting `MLEobj$control$iter.V0` to a large value, say 10. A small value will cause the algorithm to converge very slowly and 0 will generate an error. Before reporting the final log likelihood, `MARSSkem()` runs the Kalman filter with the maximum-likelihood parameter estimates (using the diffuse prior) and V0 = 0 to obtain the correct likelihood. This approach works well and is easy to implement. However when one fits a model with x0 having shared values AND an unconstrained R

matrix, an ill-conditioned matrix tends to appear in one of the steps of the Kalman filter algorithm (because V0 must have elements with 100 percent correlation if you say that some x0's have the same value). `MARSSkem()` will report warnings and errors if this happens. Switching to a diagonal R or an unconstrained x0 will fix the ill-conditioning problems. See the manual for discussion about how V0 is treated in the algorithm.

If you get errors, it generally means that the solution involves an ill-conditioned matrix. For example, your Q or R matrix is going to a value in which all elements have the same value, for example zero. If for example, you tried to fit a model with fixed and high R matrix and the variance in that R matrix was much higher than what is actually in the data, then you might drive Q to zero. Also if you try to fit a structurally inadequate model, then it is not unusual that Q will be driven to zero. For example, if you fit a model with 1 hidden state trajectory to data that clearly have 2 quite different hidden state trajectories, you might have this problem. Comparing the likelihood of this model to a model with more structural flexibility should reveal that the structually inflexible model is inadequate (much lower likelihood).

If you get a warning that the solution appears to be degenerate, it means that some of the elements in Q or R are going to zero and the log-likelihood is changing very slowly. You can either try decreasing `control$abstol` dramatically (e.g. 1E-6), use a Newton finisher `MARSSoptim(MLEobj)`, or fix the degenerate values to something very small (e.g. 1E-8) and re-estimate. Try `find.degenerate`(MLEobj) using the output from the `MARSSkem` call to find the degenerate elements.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

R. H. Shumway and D. S. Stoffer (2006). Chapter 6 in Time Series Analysis and its Applications. Springer-Verlag, New York.

Ghahramani, Z. and Hinton, G. E. (1996) Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science.

Harvey, A. C. (1989) Chapter 5 in Forecasting, structural time series models and the Kalman filter. Cambridge University Press, Cambridge, UK.

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

The EM algorithm: Holmes, E. E. (2010). Derivation of the EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models. type `show.doc(MARSS, Holmes2010)` to see.

See Also

`MARSSmcinit`, `MARSSkf`, `marssMLE`, `MARSSoptim`, `find.degenerate`

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#you can use MARSS to construct a proper marssMLE object.
MLEobj = MARSS(dat, constraint=list(Q="diagonal and equal", U="equal"), fit=FALSE)
#Use MARSS to do the fit.
kemfit = MARSSkem(MLEobj)
#see what a Newton method would find;
# wrapped in try because it tends to throw numerical errors
bfgsfit=kemfit; bfgsfit$method="BFGS"
bfgsfit = try(MARSSoptim(bfgsfit))
#look for degenerate estimates
#this will make a plot that will show if the vars converged.
#plot should be flat; if not try setting minit higher or using
#higher conv.test.slope.tol
find.degenerate(kemfit)
```

MARSSkemcheck *Model Checking for MLE objects passed to MARSSkem*

Description

This is a helper function in the MARSS-package that checks that the model can be handled by the MARSSkem algorithm.

Usage

```
MARSSkemcheck(modelObj, method="kem")
```

Arguments

modelObj	An object of class marssm.
method	The method to be used. Currently only method="kem" is available.

Value

A list with of the model elements A, B, Q, R, U, x0, Z, V0 specifying the structure of the model using text strings).

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

See Also

marssm MARSSkem

MARSSkf *Kalman Filtering and Smoothing*

Description

Implements the Kalman filter/smoother for MARSS models. This is a base function in the MARSS-package.

Usage

```
MARSSkf(y, parList, missing.matrix = NULL,
  miss.value = NULL, init.state="x10", debugkf=FALSE)
```

Arguments

y	A matrix (not dataframe), sites (rows) x years (columns). See Details regarding handling of missing values.
parList	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying parameter values. An example is the par element in a marssMLE object.
missing.matrix	Optional matrix specifying which observations are missing. See Details.
miss.value	How are missing values represented in the data? Either miss.value or missing.matrix must be supplied. If both are supplied, then miss.value will be ignored with no warning(!).
init.state	Is the initial state, x0, treated as E(x) at time t=0 (init.state="x00") or E(x) at t=1 (init.state="x10")? Default is init.state="x10" (note! for Shumway and Stoffer's Kalman filter use init.state="x00"). See Details.
debugkf	Return detailed error messages?

Details

For state space models, the Kalman filter and smoother provide optimal (minimum mean square error) estimates of the hidden states. The Kalman filter is a forward recursive algorithm which computes estimates of the states x(t) conditioned on the data up to time t. The Kalman smoother is a backward recursive algorithm which starts at time T and works backwards to t = 1 to provide estimates of the states conditioned on all data.

Missing values in the data may be handled in one of two ways: 1. Missing values may be replaced with zeroes prior to passing to MARSSkf(). Argument missing.matrix must then be a matrix of the same dimensions as the data, with 0 in the positions of observed values and 1 in the positions of missing values. 2. Data containing missing values may be passed in. Argument miss.value must then be the code used to represent missing values. The function requires that you specify either a missing matrix or a miss.value. If there are no missing values, just set miss.value to a value that is not in your data (like -99).

The expected value of the initial state, x0, is an estimated parameter (or treated as a prior). This E(initial state) can either be treated in two different ways. One can treat it as x00, meaning E(x at t=0 | y at t=0), and then compute x10, meaning E(x at t=1 | y at t=0), from x00. Or one can simply treat the initial state as x10, meaning E(x at t=1 | y at t=0). The approaches are equivalent, but the likelihood is written slightly differently in each case (sum over 2 to T versus 1 to T) and you need your likelihood calculation to correspond to how the initial state is treated in your model (either x00 or x10). The EM algorithm in the MARSS package (`MARSSkem`) follows Ghahramani's derivation and uses x10, while Shumway and Stoffer uses x00. The `init.state` argument specifies whether x00 (`init.state="x00"`) or x10 (`init.state="x10"`) is used. The default is `init.state="x10"`.

Value

A list with the following components (n is the number of state processes). Names ending in "T" are estimates from the Kalman smoother; J is also smoother output. Other components are output from the Kalman filter.

`xtT`	State estimates E[x(t) | y(1:T)] (n x T matrix).
`VtT`	State covariances E[V(t) | y(1:T)] (n x n x T array).
`Vtt1T`	Conditional error covariances E[V(t,t-1) | y(1:T)] (n x n x T).
`x0T`	Initial state mean estimates (n x 1).
`V0T`	Estimate of initial state covariance matrix (n x n).
`J`	n x n x T
`xtt1`	Forecasts E[x(t) | Y(t-1)] (n x T).
`Vtt`	State covariance estimates E[V(t) | y(1:t)] (n x n x T).
`Vtt1`	Conditional error covariances E[V(t,t-1) | y(1:t)] (n x n x T).
`Kt`	Kalman gain (n x n x T).
`Innov`	Innovations y(t) - E[y(t) | Y(t-1)] (n x T) (n x n x T).
`Sigma`	Innovations variances.
`logLik`	Log-likelihood computed from `mssm.params` and innovations.
`errors`	Any error messages due to ill-conditioned matrices.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

P. J. Brockwell and R. A. Davis (1991). Time Series: Theory and Methods.

A. C. Harvey (1989). Chapter 5, Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press.

R. H. Shumway and D. S. Stoffer (2006). Chapter 6, Time Series Analysis and its Applications. Springer-Verlag, New York.

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

See Also

`MARSS` `marssm` `MARSSkem`

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#you can use MARSS to construct a MLEobj
#MARSS calls MARSSinits to construct default initial values
MLEobj = MARSS(dat, fit=FALSE)
#Compute the kf output at the params used for the inits
kfList = MARSSkf(dat, MLEobj$start, miss.value=-99)
```

`MARSSLLprofile` *Log-likelihood profiles for MARSS Parameters*

Description

Computes log-likelihood profiles for the maximum-likelihood estimates of MARSS model parameters. This is a base function in the `MARSS-package`.

Usage

```
MARSSLLprofile(MLEobj, param=NULL, x=NULL, LLlim=3, pstep=0.01, max.steps=20, plot=
```

Arguments

`MLEobj`	An object of class `marssMLE`. Must have a `$par` element containing the MLE parameter estimates.
`param`	A vector of parameter names. Must match those output from `MARSSvectorizeparam` and output when a `marssMLE` object is printed. If you leave this off (leave NULL), then profiles will be computed for all free variables.
`x`	An optional vector of parameters values at which to comput the log-likelihood. If x is not specified then LLlim must be. If you pass in `x`, then `LLlim` and `pstep` will be ignored. If you want a different `x` for each parameter, then pass in a list: `x=list(U=c(...), Q=c(...))`. Any left off parameters will use `LLlim` and `pstep`.
`LLlim`	If passed in, the LL profile will go from max(LL)-LLlim to max(LL) to max(LL)-LLlim. The range of parameter values will keep being stepped upward or downward to reach these limits.

pstep	The value of step=parameter(x(1))-parameter(x(2)) to use when steping towards LLlim. If the parameters are from Q, R or V0, then step is on the log-scale: step=log(parameter(x(1)))-log(parameter(x(2))). If you want different step values for different parameters, pass this in as a list: `pstep=list(U=0.01, Q=.1, R=.2)`. Any parameter names left off will use the default step of 0.01.
max.steps	Since it is not guaranteed that LLlim will be reached, max.steps is used to specify the maximum number of steps to do.
plot	Whether to produce a plot. If FALSE, then just the x values and LL values are returned.

Details

Computes log-likelihood profiles for the free parameters specified in `param` for a fitted `marssMLE` object using the method `MLEobj$method`. A red line is plotted at max.LL-1.92 (corresponding to the 1 degree of freedom chi-square value) If you want to change the `control` values used for computing the log-likelihoods (to say speed things up), then change the `$control` element of `MLEobj`. If you have a lot of free parameters, then just call `MARSSLLprofile` with a few parameters at a time. Otherwise the function will try to plot all of them in one plot and you'll get a plot margin error.

Value

`MARSSLLprofile` returns a list of x-values and LL profile values for each parameter in `param`.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accesses via `show.doc(MARSS, manual)`

See Also

`MARSSparamCIs`

Examples

```
dat = t(harborSealnomiss)
dat = dat[2:3,]
kem = MARSS(dat, constraint=list(Z=factor(c(1,1)),
   R="diagonal and equal",Q=matrix(0.01)))
profLL = MARSSLLprofile(kem,c("R.1","U.1"),pstep=list(U=0.01,R=0.1))
```

marssm *Model Objects*

Description

These are model objects and utility functions for model objects in the package MARSS-package. marssm() creates multivariate autoregressive state space model objects. is.marssm() ensures model consistency. as.marssm() attempts to coerce its argument to a marssm object.

Usage

```
marssm(data = NULL, fixed, free, miss.value = -99)
is.marssm(modelObj)
as.marssm(wrapperObj)
```

Arguments

data	An optional matrix (not dataframe), sites (rows) x years (columns), of observed population abundances. If the algorithm is to be applied to log-abundance, the log transformation should be done before the data is passed in.
fixed	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying which elements of each parameter are fixed. See Details.
free	A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying which elements of each parameter are to be estimated. See Details.
miss.value	How are missing values represented in the data?
modelObj	An object of class marssm.
wrapperObj	An object of a wrapper class popWrap.

Details

A marssm object is an R representation of a MARSS model along with the data. Data for marssm() consists of multivariate time series data in which time is across columns and the n observed time series are in the n different rows.

The MARSS model is

x(t+1) = B x(t) + U + w(t), where w(t) ~ MVN(0,Q)

y(t) = Z x(t) + A + v(t), where v(t) ~ MVN(0,R)

x(1) ~ MVN(x0, V0)

In the marssm object, the MARSS model is specified by fixed/free pairs of matrices for each parameter: B, U, Q, Z, A, R, x0, V0. The dimensions for fixed and free matrices must be:

Z n x m (m<=n)

B m x m

U m x 1

Q m x m

A n x 1

R n x n

x0 m x 1

V0 m x m

The matrices in `fixed` and `free` work as pairs to specify the fixed and free elements for each parameter. In `fixed`, fixed elements must be numbers; values that are not fixed (i.e. are to be estimated) should be denoted NA. Elements in `free` will be interpreted as names for the free elements (even if they are numbers). Identical elements within a parameter matrix will be constrained to have the same value. Non-free (i.e. fixed) values should be denoted with NA, not 0, since the code will interpret 0 as "0" and assume that the user wants those parameters to be coded with the name "0" and to be estimated. See the manual (`show.doc(MARSS, manual)`) for many examples of MARSS models and their specification.

Value

An object of class "marssm".

`data`	Data supplied by user.
`fixed`	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
`free`	A list with 8 matrices Z, A, R, B, U, Q, x0, V0.
`M`	An array of dim n x n x t (an n x n missing values matrix for each time point). Each matrix is diagonal with 0 at the i,i value if the i-th value of y is missing, and 1 otherwise.
`miss.value`	Specifies missing value representation. Default is -99

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

popWrap

Examples

```
n = m = 5
fixed = free = vector("list")

free$Q=array(seq(1,m*m),dim=c(m,m)); fixed$Q=array(NA,dim=c(m,m))
free$R=array(NA,dim=c(n,n)); diag(free$R)=1
fixed$R=array(0,dim=c(n,n)); diag(fixed$R)=NA
free$Z=array(NA,dim=c(m,m))
fixed$Z=array(0,dim=c(m,m)); diag(fixed$Z)=1
free$U=array(seq(1,m),dim=c(m,1)); fixed$U = array(NA,dim=c(m,1))
fixed$A = matrix(NA,n,1); free$A = matrix(1:n,n,1)
```

```
free$B=array(NA,dim=c(m,m));
fixed$B=array(0,dim=c(m,m)); diag(fixed$B)=1
free$x0=array(seq(1,m),dim=c(m,1)); fixed$x0 = array(NA,dim=c(m,1));
free$V0=array(NA,dim=c(m,m)); fixed$V0 = array(0,dim=c(m,m))

m1 <- marssm(fixed=fixed, free=free)
is.marssm(m1)

dat = t(harborSeal)
dat = dat[2:nrow(dat),]
#allowed is a hidden variable which specifies what model structures are allowed
wrapperObj = popWrap(dat, MARSS:::allowed$kem, method="kem")
modelObj = as.marssm(wrapperObj)
```

`marssm-class` *Class "marssm"*

Description

marssm objects describe the multivariate autoregressive state space models used in the package `MARSS-package`.

Methods

print `signature(x = "marssm")`: ...

summary `signature(object = "marssm")`: ...

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

`MARSSmcinit` *Monte Carlo Initialization*

Description

Performs a Monte Carlo search for optimal initial conditions iterative maximization algorithms (`MARSSkem` and `MARSSoptim`). This is a utility function in the `MARSS-package`.

Usage

```
MARSSmcinit(MLEobj)
```

Arguments

`MLEobj`	An object of class `marssMLE`.

Details

It is recommended that initial parameter values be set using `MARSSmcinit()`, particularly if the model is not a good fit to the data. This requires more compuation time, but reduces the chance of the algorithm terminating at a local maximum and not reaching the true MLEs.

Options for `MARSSmcinit()` may be set using `MLEobj$control`, as follows:

`MLEobj$control$numInits` Number of random initial value draws.

`MLEobj$control$numInitSteps` Maximum number of EM iterations for each random initial value draw.

`MLEobj$control$boundsInits` Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn. See Examples.

The default values for these are given in `MARSSsettings.R` and listed in `MARSS`.

Value

A list with 8 matrices Z, A, R, B, U, Q, x0, V0, specifying initial values for parameters for iteration 1 of the EM algorithm.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

The user manual: Holmes, E. E. and E. J. Ward (2010) Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112 type `show.doc(MARSS, manual)` to see.

See Also

`MARSSkem` `marssMLE` `MARSS`

Examples

```
## Not run:
  #Note doing a Monte-Carlo search takes a long, long time
  dat = t(harborSeal)
  dat = dat[2:nrow(dat),]
  fit1=MARSS(dat, control=list(MCInit=TRUE))
  fit1
  #Show the inits that were used
  fit1$start
  #Try fewer initial start locations but more iterations
  fit2=MARSS(dat, control=list(MCInit=TRUE, numInits=10, numInitSteps = 100))
  fit2
  #Show the inits that were used
  fit2$start
  #ignore the values for Z,B, and V0; those parameters are fixed
```

```
## End(Not run)
```

---

marssMLE *Maximum Likelihood MARSS Estimation Object*

Description

An object in the MARSS-package that has all the elements needed for maximum-likelihood estimation of multivariate autoregressive state-space model: the data, model, estimation methods, and any control options needed for the method. If the model has been fit and parameters estimated, the object will also have the MLE parameters. Other functions add other elements to the marssMLE object, such as CIs, s.e.'s, AICb, and hessian.

Usage

```
is.marssMLE(MLEobj)
```

Arguments

MLEobj	An object of class marssMLE. See Details.

Details

The is.marssMLE() function checks components model, start and control, which must be present for estimation by functions e.g. MARSSkem. Components returned from estimation must include at least method, par, kf, numIter, convergence and logLik. Additional components (e.g. AIC) may be returned, as described in function help files.

model MARSS model specification (an object of class marssm).

start List with 7 matrices A, R, B, U, Q, x0, V0, specifying initial values for parameters to be used (if needed) by the maximization algorithm.

- B Initial value(s) for B matrix (m x m).
- U Initial value(s) for U matrix (m x 1).
- Q Initial value(s) for Q variance-covariance matrix (m x m).
- A Initial value(s) for A matrix (n x 1).
- R Initial value(s) for R variance-covariance matrix (n x n).
- x0 Initial value(s) for initial state vector (m x 1).
- V0 Initial variance(s) for initial state variance (m x m).

control A list specifying estimation options. The following options are needed by MARSSkem. Other control options can be set if needed for other estimation methods, e.g. the control options listed for optim for use with MARSSoptim. The default values for control options are set in alldefaults[[method]] which is specified in MARSSsettings.R.

- minit The minimum number of iterations to do in the maximization routine (if needed by method).

`maxit` Maximum number of iterations to be used in the maximization routine (if needed by method).

`abstol` Optional convergence tolerance for the maximization routine.

`iter.V0` The value of V0 to be used in place of 0 when x0 is treated as fixed and V0=0. See manual for discussion of initial state variance.

`trace` A positive integer. If not 0, a record will be created during maximization iterations (what's recorded depends on method of maximization).

`MCInit` If TRUE, do a Monte Carlo search of the initial condition space.

`numInits` Number of random initial value draws if MCInit=TRUE (ignored otherwise).

`numInitSteps` Number of EM iterations for each random initial value draw if MCInit=TRUE (ignored otherwise).

`boundsInits` Bounds on the uniform distributions from which initial values will be drawn if MCInit=TRUE (ignored otherwise).

`silent` Suppresses printing of progress bar, error messages and convergence information.

`silent` Suppresses printing of progress bar, error messages and convergence information.

`method` A string specifying the estimation method. MARSS 1.0 allows "kem" for Kalman-EM and "BFGS" for quasi-Newton.

`par` A list with 8 matrices of estimated parameter values Z, A, R, B, U, Q, x0, V0.

`kf` A list containing Kalman filter/smoother output.

`numIter` Number of iterations which were required for convergence.

`convergence` Convergence status and errors. 0 means converged successfully. Anything else means an error or warning.

`logLik` Log-likelihood.

Value

TRUE if no problems; otherwise a message describing the problems.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

`marssm` `MARSSkem`

marssMLE-class *Class "marssMLE"*

Description

marssMLE objects specify maximum-likelihood estimation of multivariate autoregressive state-space models in the package MARSS-package.

Methods

print `signature(x = "marssMLE")`: ...

summary `signature(object = "marssMLE")`: ...

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

MARSSoptim *Parameter estimation for MARSS models using optim*

Description

Parameter estimation for MARSS models using R's optim function. This allows access to R's quasi-Newton algorithms available via the optim function. The MARSSoptim is called when MARSS is called with `method="BFGS"`. Only diagonal Q and R matrices are allowed, if they are estimated. This is a base function in the MARSS-package. neglogLik is a helper function for MARSSoptim that returns the negative log-likelihood given a vector of the estimated parameters and a marssMLE object.

Usage

```
MARSSoptim(MLEobj)
neglogLik(x, MLEobj)
```

Arguments

MLEobj	An object of class marssMLE.
x	An vector of the estimated parameters as output by MARSSvectorizeparam.

Details

Objects of class marssMLE may be built from scratch but are easier to construct using MARSS with MARSS(..., fit=FALSE, method="BFGS").

Options for optim are passed in using MLEobj$control. See optim for a list of that function's control options. If lower and upper for optim need to be passed in, they should be passed in as part of control as control$lower and control$upper. Additional control arguments affect printing and initial conditions.

MLEobj$control$iter.V0 This is the value to which V0 will be set during the maximization algorithm (must not be equal to 0). See Details.

MLEobj$control$MCInit If TRUE, Monte Carlo initialization will be performed by MARSSmcinit.

MLEobj$control$numInits Number of random initial value draws to be used with MARSSmcinit. Ignored if control$MCInit=FALSE.

MLEobj$control$numInitSteps Maximum number of EM iterations for each random initial value draw to be used with MARSSmcinit. Ignored if control$MCInit=FALSE.

MLEobj$control$boundsInits Length 6 list. Each component is a length 2 vector of bounds on the uniform distributions from which initial values will be drawn to be used with MARSSmcinit(). Ignored if control$MCInit=FALSE. See Examples.

MLEobj$control$silent Suppresses printing of progress bars, error messages, warnings and convergence information.

Value

The marssMLE object which was passed in, with additional components:

method String "BFGS".

kf Kalman filter output.

iter.record If MLEobj$control$trace = TRUE, then this is the $message value from optim.

numIter Number of iterations needed for convergence.

convergence Did estimation converge successfully?

convergence=0 Converged in less than MLEobj$control$maxit iterations and no evidence of degenerate solution.

convergence=1 Maximum number of iterations MLEobj$control$maxit was reached before MLEobj$control$abstol condition was satisfied.

convergence=10 Some of the variance elements appear to be degenerate. T

convergence=52 The algorithm was abandoned due to errors from the "L-BFGS-B" method.

convergence=53 The algorithm was abandoned due to numerical errors in the likelihood calculation from MARSSkf. This happens frequently with "BFGS" and can sometimes be helped with a better initial condition. Try using the Kalman-EM algorithm first (method="kem"), and then using the parameter estimates from that to as initial conditions for method="BFGS".

logLik Log-likelihood.

states	State estimates from the Kalman filter.
states.se	Confidence intervals based on state standard errors, see caption of Fig 6.3 (p. 337) Shumway & Stoffer.
errors	Any error messages.

Discussion

The function only returns parameter estimates. To compute CIs, use MARSSparamCIs but if you use parametric or non-parametric bootstrapping with this function, it will use the Kalman-EM algorithm to compute the bootstrap parameter estimates! The quasi-Newton estimates are too fragile for the bootstrap routine since one often needs to search to find a set of initial conditions that work (i.e. don't lead to numerical errors).

Estimates from MARSSoptim (which come from optim) should be checked against estimates from Kalman-EM algorithm. If the quasi-Newton algorithm works, it will tend to find parameters with higher likelihood faster than the Kalman-EM algorithm. However, the MARSS likelihood surface can be multimodal with sharp peaks at degenerate solutions where a Q or R diagonal element equals 0. The quasi-Newton algorithm tends to find and gets stuck on these peaks even when they are not the maximum. Neither an initial conditions search nor starting near the known maximum (or from the parameters estimates after the Kalman-EM algorithm) will necessarily solve this problem. Thus it is wise to check against Kalman-EM estimates to ensure that the BFGS estimates are close to the MLE estimates.

Note this is mainly a problem if the time series are short or very gappy. If the time series are long, then the likelihood surface should be nice with a single interior peak. In this case, the quasi-Newton algorithm works well but it can still be sensitive (and slow) if not started with a good initial condition. Thus starting it with the estimates from the Kalman-EM algorithm is often desirable.

The V0 matrix is set to a diffuse prior if x0 is estimated (in which case V0 must be zero to treat x0 as fixed but unknown). V0 is reset to zero when the final likelihood and state estimates are computed via MARSSkf. See discussion in the manual and MARSSkem. MARSSoptim only allows diagonal Q and R matrices, if they are estimated.

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

See Also

MARSS MARSSkem marssMLE optim

Examples

```
dat = t(harborSealWA)
dat = dat[2:nrow(dat),] #remove the year row
#fit a model with 1 hidden state where obs errors are iid
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))),
   method="BFGS")

#fit a model with Kalman-EM and then use that fit as the start for BFGS
```

```
#when BFGS throws numerical errors, sometimes a close initial condition will
#fix the problem
kemfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))))
bfgsfit = MARSS(dat, constraint=list(Z=factor(c(1,1,1,1,1))),
   inits=kemfit$par, method="BFGS")
```

MARSSoptions *Change MARSS Defaults Utility*

Description

Utility in the MARSS-package to change the defaults, including default model structure, for the MARSS function.

Usage

```
MARSSoptions(..., method="kem")
```

Arguments

`...`	A name or list of names in `alldefaults`. To see what `alldefaults` looks like, type `MARSSoptions()`. This is the same as `(MARSS:::alldefaults)[[method]]`.
`method`	Estimation method. MARSS 1.0 allows method="kem" and "BFGS".

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

Examples

```
## Not run:
#Change the defaults maxit value
MARSSoptions(control=list(maxit=5000))

#Change the defaults minit, maxit and abstol value; setting abstol means the program default
MARSSoptions(control=list(minit=100, maxit=5000, abstol=0.01))

#Change lots of different defaults
MARSSoptions(control=list(minit=100, maxit=5000, abstol=0.01), constraint=list(U="unequal",Q

#Show the control defaults
MARSSoptions("control")

#Show all the defaults for method="BFGS"; this doesn't change the default method
MARSSoptions(method="BFGS")

## End(Not run)
```

MARSSparamCIs *Confidence Intervals for MARSS Parameters*

Description

Computes confidence intervals for the maximum-likelihood estimates of MARSS model parameters. This is a base function in the MARSS-package.

Usage

```
MARSSparamCIs(MLEobj, method = "hessian", alpha = 0.05, nboot=1000)
```

Arguments

MLEobj	An object of class marssMLE. Must have a $par element containing the MLE parameter estimates.
method	Method for calculating the standard errors: "hessian", "parametric", and "innovations" implemented currently.
alpha	alpha level for the 1-alpha confidence intervals.
nboot	Number of bootstraps to use for "parametric" and "innovations" methods.

Details

Approximate confidence intervals (CIs) on the model parameters may be calculated from the Hessian matrix (the matrix of partial 2nd derivatives of the parameter estimates) or parametric or non-parametric (innovations) bootstrapping using nboot bootstraps. The Hessian CIs are based on the asymptotic normality of MLE parameters under a large-sample approximation. Bootstrap estimates of parameter bias are reported if method "parametric" or "innovations" is specified.

Value

MARSSparamCIs returns the marssMLE object passed in, with additional components par.se, par.upCI, par.lowCI, par.CI.alpha, par.CI.method, par.CI.nboot and par.bias (if method is "parametric" or "innovations").

Author(s)

Eli Holmes, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov

References

Holmes, E. E. and E. J. Ward. 2010. Analysis of multivariate time-series using the MARSS package. NOAA Fisheries, Northwest Fisheries Science Center, 2725 Montlake Blvd E., Seattle, WA 98112; this is the user manual accesses via show.doc(MARSS, manual)

See Also

MARSSboot MARSSinnovationsboot MARSShessian

Examples

```
dat = t(harborSealWA)
dat = dat[2:4,]
kem = MARSS(dat, constraint=list(Z=factor(c(1,1,1)),
   R="diagonal and unequal"))
kem.with.CIs.from.hessian = MARSSparamCIs(kem)
kem.with.CIs.from.hessian
```

MARSSsimulate *Simulate Data from a MARSS Model and Parameter Estimates*

Description

Generates simulated data from a MARSS model with specified parameter estimates. This is a base function in the MARSS-package.

Usage

```
MARSSsimulate(parList, tSteps = 100, nsim = 1, silent = TRUE,
  miss.loc = NULL, miss.value = NULL)
```

Arguments

parList	A list of parameter matrices structured like the $par element of an object of class marssMLE.
tSteps	Number of time steps in each simulation.
nsim	Number of simulated data sets to generate.
silent	Suppresses progress bar.
miss.loc	Optional matrix specifying where to put missing values. See Details.
miss.value	Code representing missing values in miss.matrix. See Details.

Details

Argument miss.loc is an array of dimensions n x tSteps x nsim, specifying where to put missing values in the simulated data. Locations where missing data appear are specified using the miss.value; non-missing values can be specified by any other numeric value. If the locations of the missing values are the same for all simulations, miss.loc can be a matrix of dim=c(n, tSteps) (the original data for example). If miss.loc is passed in, miss.value must be specified. The default is that there are no missing values. If one's data has missing values in it and one want to replicate those locations in the simulated data, miss.loc can simply be set to the original data (see examples).

Value

sim.states	Array (dim m x tSteps x nsim) of state processes simulated from parameter estimates.
sim.data	Array (dim n x tSteps x nsim) of data simulated from parameter estimates.
par	The list of parameter matrices from which the data were simulated.
miss.loc	Matrix identifying where missing values are located.
tSteps	Number of time steps in each simulation.
nsim	Number of simulated data sets generated.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

See Also

marssm marssMLE MARSSboot

Examples

```
#do a parametric bootstrap.
#Same length as original data and same location of missing data
d = harborSeal[,2:4]
dat = t(d)
MLEobj = MARSS(dat)
sim.obj = MARSSsimulate(parList=MLEobj$par, tSteps=dim(d)[1], nsim=10)
```

MARSSvectorizeparam

Vector to Parameter Matrix Conversion

Description

Converts MLEobj$par to a vector of the estimated parameter elements and vice versa. This is a utility function in the MARSS-package.

Usage

```
MARSSvectorizeparam(MLEobj, parvec = NA)
```

Arguments

MLEobj	An object of class marssMLE.
parvec	NA or a vector. See Value.

Details

Utility function to generate parameter vectors for optimization functions, and to set `MLEobj$par` using a vector of parameter values (only the estimated values).

Value

If parvec=NA, a vector of estimated parameters. Otherwise, a `marssMLE` object with `$par` set by parvec.

Author(s)

Eli Holmes and Kellie Wills, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

See Also

`marssMLE`

Examples

```
dat = t(harborSealWA)
dat = dat[2:3,]
kem = MARSS(dat)
paramvec = MARSSvectorizeparam(kem)
paramvec
```

---

plankton *Plankton Data Sets*

Description

Example data sets for use in MARSS vignettes for the `MARSS-package`. The "lakeWAplankton" plankton counts have been standardized to a mean of zero and variance of 1 (Z-score transformation). The second column in "lakeWAplankton" is simply a index for the count. Columns 3 and 4 are month and month^2 but they have also been Z-score transformed. Since MARSS functions require time to be across columns, these data matrices must be transposed before passing into MARSS functions.

Usage

```
data(ivesDataLP)
data(lakeWAplankton)
```

Format

The data are provided as a matrix with time running down the rows.

Source

- ivesDataLPIves, A. R. Dennis, B. Cottingham, K. L. Carpenter, S. R. (2003) Estimating community stability and ecological interactions from time-series data. Ecological Monographs, 73, 301-330.
- lakeWAplanktonHampton, S. E. Scheuerell, M. D. Schindler, D. E. (2006) Coalescence in the Lake Washington story: Interaction strengths in a planktonic food web. Limnology and Oceanography, 51, 2042-2051.

Examples

```
str(ivesDataLP)
str(lakeWAplankton)
```

---

popWrap *Wrapper Objects*

Description

Wrapper objects are used by the function MARSS in the package MARSS-package. popWrap creates a wrapper object containing specifications and options for estimation of a multivariate autoregressive state-space model.

Usage

```
popWrap(y, allowed,
    inits=NULL,
    constraint=NULL,
    fixed=NULL, free=NULL,
    miss.value=NULL,
    control=NULL,
    method=NULL,
    silent=FALSE)
```

Arguments

y	A matrix (not dataframe), observations (rows) x time steps (columns), of data.
allowed	Allowed constraints. This is determined by the method used for parameter estimation. allowed$kem (specified in MARSSsettings is the set of allowed constraints for the Kalman-EM algorithm under MARSS 1.0. allowed$BFGS are the allowable constraints for BFGS.
inits	List with up to 8 matrices Z, A, R, B, U, Q, x0, V0, specifying initial values for parameters to use in iterative ML estimation algorithms, such as Kalman-EM and quasi-Newton methods. These are ignored if the specified parameter is not being estimated. If not passed in by the user, MARSS creates generic inits using MARSSinits.

B Initial value(s) for B parameter (length 1 or m x m). If length 1, inits$B constructed as diag(value, m).

U Initial value(s) for U parameter (length 1 or m x 1). If length 1, inits$U constructed as matrix(value, nrow=m, ncol=1).

Q Initial value(s) for Q parameter (length 1 or m x m). If length 1, inits$Q constructed as diag(value, m).

Z Initial value(s) for Z parameter (n x m). Ignored in MARSS 1.0; included for MARSS 2.0.

A Initial value(s) for A parameter (length 1 or n x 1). If length 1, inits$U constructed as matrix(value, nrow=n, ncol=1).

R Initial value(s) for R parameter (length 1 or n x n). If length 1, inits$R constructed as diag(value, n).

x0 Initial value(s) for x0 parameter (length 1 or m x 1). If length 1, inits$x0 constructed as matrix(value, nrow=m, ncol=1).

V0 Initial variance(s) for hidden states (length 1 or m x m). Ignored in MARSS 1.0; included for forward compatibility.

constraint Model specification using parameter constraint descriptions (recommended). See MARSS for details.

fixed Optional model specification using matrices of fixed and free parameters. See manual for details.

free Optional model specification using matrices of fixed and free parameters. See manual for details.

miss.value How are missing values represented in the data?

method The method used for estimation. This is needed for setting default values for control.

control Control options for maximization algorithms.

minit Minimum number of EM iterations.

maxit Maximum number of EM iterations.

abstol Optional tolerance for log-likelihood change. If log-likelihood changes less than this amount relative to the previous iteration, the algorithm exits.

iter.V0 This is the value to which V0 will be set during the algorithm iterations (!=0). See MARSSkem.

trace Positive integer. If not 0, a record will be created of each variable over the maximization iterations. The information recorded depends on the maximization method.

safe If TRUE, MARSSkem will rerun MARSSkf after each individual parameter update rather than only after all parameters are updated.

MCInit Use Monte Carlo initialization? See discussion in MARSSkem and MARSSmcinit.

numInits Number of random initial value draws.

numInitSteps Number of EM iterations for each random initial value draw.

boundsInits Bounds on the uniform distributions from which initial values will be drawn. Note that bounds for the covariance matrices Q and R, which require positive values, are specified in logs.

silent Suppresses printing of progress bars, error messages, warnings and convergence information.

Details

Wrapper functions e.g. MARSS call popWrap() to create a 'popWrap' object, then is.marssm to coerce this object to class 'marssm' for the estimation function. The popWrap() function calls checkPopWrap to check user inputs.

If arguments inits, constraint, or control are not provided by the user, they will be set by the alldefaults[[method]] object specified in MARSSsettings. Argument constraint is a convenient way to specify model structure for certain common cases. If fixed is included, it provides matrices for some parameters, these will override any constraints for those parameters. See marssm or the manual for instructions on how to specify fixed and free matrices.

Value

An object of class 'popWrap'.

data	Data supplied by user.
m	Number of hidden state trajectories.
constraint	A list with up to 8 elements Z, A, R, B, U, Q, x0, V0 (unless some of these are specified in "fixed"). See MARSS for details on what values are allowed.
fixed	A list with (up to) 8 matrices Z, A, R, B, U, Q, x0, V0.
free	A list with (up to) 8 matrices Z, A, R, B, U, Q, x0, V0.
inits	A list specifying initial values for parameters to be used at iteration 1 in iterative maximum-likelihood algorithms.
miss.value	Specifies missing value representation.
method	The method used for estimation.
control	See Arguments.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

See Also

MARSS marssm checkPopWrap as.marssm

Examples

```
dat = t(harborSeal)
dat = dat[2:nrow(dat),]
wrapperObj = popWrap(dat, allowed=MARSS:::allowed$kem, method="kem")
```

popWrap-class *Class "popWrap"*

Description

popWrap objects are wrapper object containing specifications and options for estimation of multivariate autoregressive state-space models in the package MARSS-package.

Author(s)

Kellie Wills, NOAA, Seattle, USA.

kellie(dot)wills(at)noaa(dot)gov

show.doc *Documentation Utility*

Description

Utility in the MARSS-package to open documentation for R packages. This is used to open files and show the directory of the doc directory of a package. This is similar but a bit more flexible than that R utility RShowDoc.

Usage

```
show.doc(pkg, filename, dir="doc")
```

Arguments

pkg	The name of an R package. Need not be in quotes.
filename	A file name or "manual", "index" or "dir". Need not be in quotes. Using "manual", "index", and "dir" has special behavior (no quotes required). "manual" opens (tries to open) the file manual.pdf. "index" tries to open the file index.html. "dir" lists the files in the directory (specified by the dir argument).
dir	The package subdirectory in which the file is located. Typically documentation is in the doc subdirectory and this is the default. However, other package directories can be specified. Use dir="." to see the directories available.

Author(s)

Eli Holmes eli(dot)holmes(at)noaa(dot)gov

See Also

See Also RShowDoc

Examples

```
## Not run:
#MARSS is used as the package in these examples, but you can replace
#it with any package name you have loaded into R.  show.doc is not
#MARSS specific

#list the files in the "doc" subdirectory
show.doc(MARSS, dir)
#show the file manual.pdf in the "doc" subdirectory
show.doc(MARSS, manual)
#show the index.html file in the "doc" subdirectory
show.doc(MARSS, index)
#Open the file Case_study_2.R in the "doc" subdirectory
show.doc(MARSS, Case_study_2.R)
#show the subdirectories in the package directory
show.doc(MARSS, dir, dir=".")

## End(Not run)
```

---

stdInnov *Standardized Innovations*

Description

Standardizes Kalman filter innovations. This is a helper function called by MARSSinnovationsboot in the MARSS-package.

Usage

```
stdInnov(SIGMA, INNOV)
```

Arguments

SIGMA	n x n x T array of Kalman filter innovations variances. This is output from MARSSkf.
INNOV	n x T matrix of Kalman filter innovations. This is output from MARSSkf.

Details

n = number of observation (y) time series. T = number of time steps in the time series.

Value

n x T matrix of standardized innovations.

Author(s)

Eli Holmes and Eric Ward, NOAA, Seattle, USA.

eli(dot)holmes(at)noaa(dot)gov, eric(dot)ward(at)noaa(dot)gov

References

Stoffer, D. S., and K. D. Wall. 1991. Bootstrapping state-space models: Gaussian maximum likelihood estimation and the Kalman filter. Journal of the American Statistical Association 86:1024-1033.

See Also

MARSSboot MARSSkf MARSSinnovationsboot

Examples

```
  ## Not run:
  std.innovations = stdInnov(kfList$Sigma, kfList$Innov)

## End(Not run)
```

Index

∗Topic **classes**
marssm-class, 39
marssMLE-class, 43
popWrap-class, 54
∗Topic **datasets**
graywhales, 8
harborSeal, 9
loggerhead, 11
plankton, 50
∗Topic **hplot**
CSEGriskfigure, 5
CSEGtmufigure, 6
∗Topic **package**
MARSS-package, 2

alldefaults *(allowed)*, 3
allowed, 3
as.design *(is.blockdiag)*, 10
as.marssm, *4*, *53*
as.marssm *(marssm)*, 37

checkPopWrap, *3*, 4, *53*
CSEGriskfigure, 5, *7*
CSEGtmufigure, 6, *6*

describe.marssm *(allowed)*, 3

fdHess, *24*
find.degenerate, 7, *31*

graywhales, 8
grouse *(graywhales)*, 8

harborSeal, 9
harborSealnomiss *(harborSeal)*, 9
harborSealWA *(harborSeal)*, 9

Imat *(is.blockdiag)*, 10
is.blockdiag, 10
is.blockequaltri *(is.blockdiag)*, 10
is.blockunconst *(is.blockdiag)*, 10
is.design *(is.blockdiag)*, 10
is.diagonal *(is.blockdiag)*, 10
is.equaltri *(is.blockdiag)*, 10
is.fixed *(is.blockdiag)*, 10
is.identity *(is.blockdiag)*, 10
is.marssm, *53*
is.marssm *(marssm)*, 37
is.marssMLE *(marssMLE)*, 41
is.wholenumber *(is.blockdiag)*, 10
isleRoyal *(graywhales)*, 8
ivesDataLP *(plankton)*, 50

kem.methods *(allowed)*, 3

lakeWAplankton *(plankton)*, 50
loggerhead, 11
loggerheadNoisy *(loggerhead)*, 11

makediag *(is.blockdiag)*, 10
MARSS, *2–4*, *8*, 12, *27*, *28*, *35*, *40*, *43–45*, *51–53*
MARSS-package, *4–12*, *14*, *16*, *18*, *20*, *21*, *23–26*, *28*, *32*, *33*, *35*, *37*, *39*, *41*, *43*, *46–51*, *54*, *55*
MARSS-package, 2
MARSSaic, *3*, 18, *22*
MARSSapplynames, 20
MARSSboot, *3*, *6*, *19*, 21, *27*, *48*, *49*, *56*
MARSScheckdims, 23
MARSScheckpar *(MARSScheckdims)*, 23
MARSShessian, 24, *48*
MARSSinits, 25, *51*
MARSSinnovationsboot, 26, *48*, *55*, *56*
MARSSkem, *2*, *8*, *16*, *24–27*, 28, *32*, *34*, *35*, *39–42*, *45*, *52*
MARSSkemcheck, 32
MARSSkf, *2*, *8*, *16*, *29–31*, 33, *44*, *45*, *52*, *55*, *56*
MARSSLLprofile, 35

marssm, *12*, *14–16*, *20–22*, *25, 26*, *32*, *35*, 37, *41, 42*, *49*, *53*
marssm-class, 39
MARSSmcinit, *29*, *31*, 39, *44*, *52*
marssMLE, *4–6*, *12*, *16*, *18–22*, *24*, *27–29*, *31*, *33*, *35, 36*, *39, 40*, 41, *41*, *43–45*, *47–50*
marssMLE-class, 43
MARSSoptim, *2, 3*, *16*, *25, 26*, *31*, *39*, *41*, 43
MARSSoptions, 46
MARSSparamCIs, *3*, *5*, *24*, *27*, *36*, *45*, 47
MARSSsettings, *15*
MARSSsettings *(MARSS)*, 12
MARSSsimulate, *2*, 48
MARSSvectorizeparam, *35*, *43*, 49
model.elem *(allowed)*, 3

neglogLik *(MARSSoptim)*, 43
nlme, *24*

optim, *3*, *41*, *43–45*
optim.methods *(allowed)*, 3

plankton, 50
popWrap, *4*, *37, 38*, 51
popWrap-class, 54
prairiechicken *(graywhales)*, 8
print,marssm-method *(marssm-class)*, 39
print,marssMLE-method *(marssMLE-class)*, 43

RShowDoc, *54*

show.doc, 54
stdInnov, *27*, 55
summary,marssm-method *(marssm-class)*, 39
summary,marssMLE-method *(marssMLE-class)*, 43

takediag *(is.blockdiag)*, 10

unvec *(is.blockdiag)*, 10

vec *(is.blockdiag)*, 10

wilddogs *(graywhales)*, 8